[illegible]

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/903,019	07/10/2001	Murari Kumar	42390P11769	9066

8791 7590 10/05/2004

BLAKELY SOKOLOFF TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1030

EXAMINER	
CHOW, CHIH CHING	
ART UNIT	**PAPER NUMBER**
2122	

DATE MAILED: 10/05/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

Office Action Summary	Application No.	Applicant(s)
	09/903,019	KUMAR, MURARI
	Examiner	Art Unit
	Chih-Ching Chow	2122

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *10 July 2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-22* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-22 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *10 July 2001* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. ______.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. ______ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application (PTO-152)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.	6) ☐ Other: _____.

DETAILED ACTION

1. This action is responseive to the application filed on July 10, 2001.
2. The priority date considered for this application is July 10, 2001.
3. Claims 1-22 have been examined.

Drawings

4. The drawings are objected because of the following minor informalities, any required corrective action should be made in the next Office action. The objection to the drawings will not be held in abeyance.

   a. Figure 11 should be designated by a legend such as --Prior Art-- because only that which is old is illustrated. See MPEP § 608.02(g). See Tom Fout "Universal Plug and Play in Windows XP', page 6, Figure 1.

   b. 'HPP' in 120 of FIG. 2 is not described in specification. It's not clear what is the function of it.

Claim Objections

5. Claim 19 is objected to because of the following informalities: claim didn't end with a period; appropriate correction is required.

6. Claim 2 is objected to because of the following informalities: claim 1 cited '**service control** class files', however claim 2 referred '**service-control** class files'; examiner assume they meant to be the same thing, however they should be referred in a consistent name though out the entire document; appropriate corrections are required.

7. The current invention contains lots of implementation details, which are irrelevant to the current invention. The inventor specifically mentioned in paragraph 32, "In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without some of these specific details. .... these examples should not be construed in a limiting sense as they are merely intended to provide examples of the present invention rather than to provide an exhaustive list of all possible implementations of the present invention." Examiner eliminated all the design choices from the claim languages, and looked into the 'concepts' behind the implementation details. The claims should be corrected to encompass only the concepts, not the implementation details.

Claim Rejections - 35 USC § 103

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

9. Claims 1, 2, 4-6, 8-11, 13-15, 17-22 are rejected under 35 U.S.C. 103(a) as being unpatentable over US 2002/0112058, by Weisman et al. (hereinafter "Weisman"), in view of US patent No. 6,549,943 by Maximilian J. Spring (hereinafter "Spring"), and further in view of 'UPnP Device Architecture', June 2000 (hereinafter "UPnP").

CLAIM	**Weisman / Spring / UPnP**
1. A method comprising: (a) receiving a UPnP device description document from a device developer;	For item a, in Weisman, paragraph 1004, "retrieving the **UPnP description for a device** is simple: the **control point** issues an HTTP GET request on the URL in the discovery message, and the device returns the description document."; thus a UPnP device description document is received,
(c) receiving the service control class files including updated service-control stub-methods modified by the device developer for responding to actions and events received by a UPnP device described by the UPnP device description document; and	For item c, in paragraph 1005, "UPnP vendors can **differentiate their devices** by extending services, including additional UPnP services, or embedding additional UPnP devices (***updated services***). When a control point **retrieves** a particular device's description, these added features are exposed to the control point for control, eventing, and presentation.", paragraph 1008, "The UPnP description for a device contains several pieces of vendor-specific information, definitions of embedded devices and services, and URLs for control, eventing, and presentation of the device." therefore an **updated device description or modification responds to actions and events** can also be received.
(b) generating one or more service control class files including one or more service-control stub-methods;	For items b, in Weisman, claim 9, "A computer-readable data carrying medium having a **link-able program module** thereon, the **program module executable** on a computer in a network **of computing devices** interoperating via a peer networking protocol to provide hosting of the peer networking protocol for **logical device software** that operates as a logical device having **a set of services** (***Service-control class files and stub-methods***) on the computer," – Weisman teaches the concept of specific service(s) provided by a certain device has already been generated and sent to a UPnP environment for user to use, but Weisman does not mention 'generating' and 'compiling' service control classes. However, Spring teaches in an analogous art, in Spring, claim 1, "A method

of computing and storing a result value used to describe a device in **a tree representation** in a network management system, the method comprising the computer-implemented steps of: creating and storing an **abstract textual description of a network device**,…. **compiling** the abstract textual description into one or more program source code files, based on a language specification of a grammar for the abstract textual description; automatically **compiling the program source code files** to result in **generating executable** code for the network management system".
It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement of the Weisman's UPnP device description disclosure with the compiling and generating executables further taught by Spring for the purpose of creating and storing the executables for a containment tree structured network devices (see Spring Abstract, lines 8-10).

(d) compiling the service-control class files and the updated service-control stub-methods along with a device class library and an UPnP software development kit to generate a device executable for the UPnP device described by the UPnP device description document.

For item d, in Weisman, paragraph 46, "Alternative implementations of the Device Host can be structured with various other software architectures, such as with Device Host operations and interaction implemented in different configurations of **executable programs, library modules,** processes, objects, programming interfaces and **procedure calls**." – the new device's services executable programs would have to be generated and included in the new configuration; therefore a compiling job would have to be done in order to get a new executable program to be created.
A further example is in another analogous art, in UPnP, page 20, "**actionList** Required if and only if the **service** has **actions**. (Each service may have >= 0 actions.) Contains the following sub

element(s):
action Required. Repeat once for each action defined by a UPnP Forum working committee. If UPnP vendor **differentiates service** by adding additional actions, repeat once for each additional action. Contains the following sub elements:
name Required. Name of action. Must not contain a hyphen character (-, 2D Hex in UTF-8) nor a hash character (#, 23 Hex in UTF-8).

- For standard actions defined by an UPnP Forum working committee, must not begin with **X_** nor **A_**.
- For non-standard actions specified by an UPnP vendor and added to a standard service, must begin with **X_**. String. Should be < 32 characters."

In order for any of the action/command to work, the updated services would have to be **recompiled**, linked and a new executable (action) thus can be created. Therefore UPnP has further taught us generating new service action, **compiling** the updated code and generating a new executable for the UPnP device described by the **UPnP device description document**.

It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement of the Weisman's disclosure with the further taught by UPnP for the purpose of supporting zero-configuration, "invisible" networking, and automatic discovery for a breadth of device categories from a wide range of vendors. This means a device can dynamically join a network, obtain an IP address, convey its capabilities, and learn about the presence and capabilities of other devices (see UPnP, 2nd paragraph).

2. The method of claim 1, wherein generating the service-control class files further comprises: (a) parsing the UPnP device description document to determine a root device including one or more services and one or more embedded device each including one or more services, each service defined by a service control protocol description (SCPD) file;	For the features of claim 1 see claim 1 rejection. For item a, in Weisman, paragraph 54, "A **UPnP device description** is an XML document that describes the **properties** of a device and the hierarchy of **nested devices** within it.", and paragraph 854, "When a device is added to the network, it multicasts discovery messages to advertise its **root device**, to advertise any **embedded devices**, and to advertise its **services**." Weisman teaches that there may be one or more services and one or more embedded device (***nested devices***). Further, in paragraph 1155 -1156, and 1077-1078, Weisman discloses the device description document format for a specific device. " [1159] **scpd** [1160] Has urn:schemas-upnp-org:service:1:0 as the value for the xmlns attribute; this references the UPnP Template Language (explained below). Case sensitive. Contains all other elements describing the service". [1077] **SCPD**URL [1078] URL for service description (**Service Control Protocol Definition** URL). (cf. section on Control.) May be relative to base URL. Specified by UPnP vendor.
(b) generating a service-control class file for each of the one or more services of the root device; (c) selecting an embedded device from the one or more embedded devices of the root device; (d) generating a service-control class file for each of the one or more services of the selected embedded device; (e) repeating the generating and	For items b-f, Weisman discloses repeating the generating for each of the services available in environment, see paragraph 1071 to 1072, " [1071] **service** [1072] **Repeated once for each service** defined by a UPnP Forum working committee. If UPnP vendor differentiates device by adding additional, standard UPnP services, **repeat**ed once for

selecting for each of the one or more embedded devices of the root device; and

(f) once each of the service control class files are generated, generating a class service linker file based on the generated service-control classes, the class service file linker enabling linking the service-control classes during compiling to generate the device executable.

additional service."
Again, both Weisman and Spring discloses the device executable feature (***program module executable, link-able program module, compiling and generating new executable***), see claim 1 (b) rejection.

4. The method of claim 2, wherein generating service control class files for the root device further comprises:
(a) selecting a service from the one or more services of the root device;
(b) generating a class generator object for the selected service of the root device;

For the features of claim 2 see claim 2 rejection. For item a, b and f, in Spring, column 6, lines 9-15, "The ADD language also comprises language constructs that may be used to define features of a managed device. The functional definitions that are reflected in the definition constructs are written in a **Java-like syntax** with extensions. Embedded within functional definitions, an **object** designator syntax (***class generator object for the selected service of the root device***) is used to specify references to values coming from the device." Item b is actually a design choice step.

(c) passing an SCPD file describing the selected service, including a device identification code of the root device and a service identification of the selected service to the class generator object;
(d) generating, by the class generator object, a service control class file based on the received SCPD file, the service identification code and the device identification code;

Items c and d. In Weisman, paragraphs 1048 and 1049,
"[1048] **UDN**

[1049] **Universal Device Name**. Universally-unique **identifier for the device**, whether root or embedded." UPnP further teaches, on page 42, 3rd paragraph, "A device description contains (among other things), for each service, an eventing URL (in the eventSubURL element) and a **service identifier** (in the serviceId element); these correspond to the URL and service identifier for the publisher, respectively." UPnP also teaches the 'header file' (item e), under page 7, section

(e) generating a header file

corresponding to the generated service control class file and the SCPD file; (f) once generated, destroying the class generator object; and (g) repeating the selecting, generating, passing, generating and destroying for each of the one or more services of the root device.	1.1.2., "When a device is added to the network, it multicasts discovery messages to advertise its **root device**, to advertise any embedded **devices**, and to advertise its **services**. Each discovery message contains four major components: 1. a potential search target (e.g., device type), sent in an **NT header**, 2. a composite identifier for the advertisement, sent in a **USN header**, 3. a URL for more information about the device (or enclosing device in the case of a service), sent in a **LOCATION header**, and 4. a duration for which the advertisement is valid, sent in a **CACHE-CONTROL header**." And further teaches the 'destroying ' for unused note, under section 1.1.3, "if not re-advertised, the discovery message eventually expires on its own and must be **removed** from any control point cache". Item g, 'repeating' has been cited in claim 2 rejection.
5. The method of claim 2, wherein generating service control class files for the selected embedded device further comprises: (a) selecting a service from the one or more services of the selected embedded device; (b) generating a class generator object for the selected service of the selected embedded device; (c) passing an SCPD file describing the selected service, including a device identification code of the selected embedded device and a service identification code of the selected service to the class generator object; (d) generating, by the class generator object, a service control class file based on the received SCPD file, the	For the features of claim 2 see claim 2 rejection. For the rest of the features in claim 5, see claim 4 rejection (the process repeated the same way for both root device and embedded devices, see claim 2 rejection).

service identification code and the device identification code;
(e) generating a header file corresponding to the generated service control class file and the SCPD file;
(f) once generated, destroying the class generator object; and
(g) repeating the selecting, generating, passing, generating and destroying for each of the one or more services of the selected embedded device.

6. The method of claim 2, wherein generating the service linker class file further comprises:
(a) generating a class service linker object;
(b) storing class information regarding each generated service control class file within a service table of the class service linker object;
(c) generating, by the class service linker object, the class service linker file based on the service table; and
(d) destroying the class service linker object.

For the features of claim 2 see claim 2 rejection. In Spring, column 6, lines 46-53, "At **compilation** time, one or more MIB files 220 are provided to a package **linker** 218. The MIB files contain pre-determined definitions of MIBs that are contained in the real-world network device that is represented by ADD 202. As output, package **linker** 218 produces a package descriptor 222 (***generating and storing***)"; the concept of using linker to access data is cited here, the design choice (language specific detail) is up to vendors.

8. The method of claim 1, further comprising:
(a) executing the device executable;
(b) creating an instance of the root device and each of the one or more services of the root device;
(c) creating an instance of each embedded device and each of the one or more services of the respective embedded device;
(d) organizing the root device and embedded devices, as well as the services of the root device and the embedded devices within a tree hierarchy based on the device

For the features of claim 1 see claim 1 rejection. For items a-c, see claim 2 rejection. For item d, in Spring, claim 1, "A method of computing and storing a result value used to describe a device in **a tree representation** in a network management system". For item e, in Weisman, paragraph 6, "The hosted devices **register** with the device hosting framework by providing information about their properties....The hosted devices also **register service objects** with the device hosting framework for **each service** they provide that is to be controllable through the peer networking description document to form a device object tree; and

(e) registering the root device and the one or more embedded device within the device object tree with the device class library to enable receipt of events for the services of the root device and the services of the one or more registered embedding devices.

protocol."; paragraph 47, "This enables the device/bridge developer to **expose** the hosted **devices** and their services through the peer networking protocol by writing the hosted device program as a **library** or **executable** to **register** and interface with the Device Host for hosting by the Device Host, and avoid having to fully implement the peer networking protocol individually in each hosted device.".

9. The method of claim 8, wherein registering the root device and one or more embedded devices further comprises:

(a) registering an event listener object of the device class library with the UPnP software development kit to enable receipt of action and event requests received/generated by one or more control points of the UPnP device;

(b) registering the root device and one or more embedded devices with the event listener object;

(c) receiving, by the event listener object, a respective action/event request from a control point;

(d) finding a service object for response to the respective action request using a received device identification code and service identification code;

(e) once the service object is found, invoking a callback function of the service object to determine an appropriate action method to execute in response to the respective action request;

(f) executing the appropriate action method; and

(g) once the action method is processed, setting an event object with a response string that is received by

For the features of claim 8 see claim 8 rejection. For item b see Spring, in claim 8 rejection. For item a, in Weisman, paragraph 7, "The device hosting framework also **listens** for **control requests** (***event listener***) in the peer networking protocol that are targeted at the hosted devices and services (item c, ***receives request***). The device hosting framework translates the control requests into **calls to the service objects' programming interfaces** (items d, e, f, ***finding and invoking/executing action or event***). The device hosting framework also translates the return information from the programming interface methods into valid **control responses** (item g, ***callback function and response string***) in the peer networking device **control** protocol.", Under UPnP, page 5, 1. 'Discovery', "Discovery is Step 1 in UPnP networking. Discovery comes after addressing (Step 0) where devices get a network address. Through discovery, control points find interesting device(s). Discovery enables description (Step 2) where control points learn about device capabilities, control (Step 3) where a **control point sends commands to device**(s), eventing (Step 4) where control points **listen** to state changes in device(s), and presentation (Step 5) where control points display a user interface for device(s)." and page 6, 3rd paragraph, "All

the control point.	devices must **listen** to the standard multicast address for these messages and must **respond** if any of their embedded devices or services match the search criteria in the discovery message." The 'callback' is also disclosed under '4.1.1 Eventing: Subscribing: SUBSCRIBE with NT and CALLBACK' on page 42.
10. A computer readable storage medium including program instructions that direct a computer to function in a specified manner when executed by a processor, the program instructions comprising: receiving a UPnP device description document from a device developer', generating one or more service control class files including one or more service-control stub-method; receiving the service control class files including updated service-control stub-methods modified by the device developer for responding to actions and events received by a UPnP device described by the UPnP device description document; and compiling the service-control class files and the updated service-control stub-methods along with a device class library and a UPnP software development kit to generate a device executable for the UPnP device.	Same as claim 1 rejection.
11. The computer readable storage medium of claim 10, wherein generating the service-control class files further comprises: parsing the UPnP device description document to determine a root device including one or more services and one or more embedded device each including one or more services, each service defined by a	For the features of claim 10 see claim 10 rejection. For the rest of the features in claim 11, see claim 2 rejection.

service control protocol description (SCPD) file;
generating a service-control class file for each of the one or more services of the root device;
selecting an embedded device from the one or more embedded devices of the root device;
generating a service-control class file for each of the one or more services of the selected embedded device;
repeating the generating and selecting for each of the one or more embedded devices of the root device; and
once each of the service control class files are generated, generating a class service linker class file based on the generated service-control classes, the class service linker file enabling linking the service-control classes during compiling to generate the device executable.

13. The computer readable storage medium of claim 11, wherein generating service control class files for the root device further comprises:
selecting a service from the one or more services of the root device;
generating a class generator object for the selected service of the root device;
passing an SCPD file describing the selected service, including a device identification code of the root device and a service identification of the selected service to the class generator object;
generating, by the class generator object, a service control class file based on the received SCPD file, the service identification code and the device identification code;

For the features of claim 11 see claim 11 rejection. For the rest of the features see claim 4 rejection.

generating a header tile corresponding to the generated service control class file and the SCPD file; once generated, destroying the class generator object; and repeating the selecting, generating, passing, generating and destroying for each of the one or more services of the root device.	
14. The computer readable storage medium of claim 11, wherein generating service control class files for the selected embedded device further comprises: selecting a service from the one or more services of the selected embedded device; generating a class generator object for the selected service of the selected embedded device; passing an SCPD file describing the selected service, including a device identification code of the selected embedded device and a service identification code of the selected service to the class generator object; generating, by the class generator object, a service control class file based on the received SCPD file, the service identification and the device identification; generating a header file corresponding to the generated service control class file and the SCPD file; once generated, destroying the class generator object; and repeating the selecting, generating, passing, generating and destroying for each of the one or more services of the selected embedded device.	For the features of claim 11 see claim 11 rejection. For the rest of the features see claim 5 rejection.

15. The computer readable storage medium of claim 11, wherein generating the service linker class file further comprises: generating a class service linker object; storing class information regarding each generated service control class file within a service table of the class service linker object; generating, by the class service linker object, the service class linker file based on the service table; and destroying the class service linker object.	For the features of claim 11 see claim 11 rejection. For the rest of the features see claim 6 rejection.
17. The computer readable storage medium of claim 10, further comprises: executing the device executable; creating an instance of the root device and each of the one or more services of the root device; creating an instance of each embedded device and each of the one or more services of the respective embedded device; organizing the root device and embedded devices, as well as the services of the root device and the embedded devices within a tree hierarchy based on the device description document to form a device object tree; and registering the root device and one or more embedded devices of the device object tree with the device class library to enable receipt of actions/events for the one or more services of the root device and the one or more services of the one or more registered embedded devices.	For the features of claim 10 see claim 10 rejection. For the rest of the features see claim 8 rejection.
18. The computer readable storage medium of claim 17, wherein	For the features of claim 17 see claim 17 rejection. For the rest of the features see

registering the root device and one or more embedded devices further comprises: registering an event listener object of the device class library with the UPnP software development kit to enable receipt of action and event requests received/generated by one or more control points of the UPnP device; registering the root device and one or more embedded devices with the event listener object; receiving, by the event listener object, a respective action/event request from a control point; finding a service object for response to the respective action request using a received device identification code and service identification code; once the service object is found, invoking a callback function of the service object to determine an appropriate action method to execute in response to the respective action request; executing the appropriate action method; and once the action method is processed, setting an event object with a response string that is received by the control point.	claim 9 rejection.
19. An apparatus, comprising: a processor having circuitry to execute instructions; a communications interface coupled to the processor, the communications interface to advertise services to a control point, provide device description to the control point, provide service description for each service to the control point, to receive action/event requests from the control point and to publish updates during	Same as claim rejection 1and claim 9 rejection (***response to actions/events***).

state changes in response to received action/event requests; and a storage device coupled to the processor, having sequences of instructions stored therein, which when executed by the processor cause the processor to: receive the service control class files including updated service-control stub-methods modified by the device developer for responding to actions and events received by a UPnP device described by the UPnP device description document, compile the service-control class files and the updated service-control stub-methods along with a device class library and a UPnP software development kit to generate a device executable for the UPnP device described by the UPnP device description document, and execute the device executable to enable response to actions/events received by the UPnP device	
20. The apparatus of claim 19, wherein the instruction to execute the device executable further causes the processor to: create an instance of a root device and each of one or more services of the root device; create an instance of each embedded device of the root device and each of one or more services of a respective embedded device; organize the root device and embedded devices, as well as the services of the root device and the embedded devices within a tree hierarchy based on the device description document to form a device object tree;	For the features of claim 19 see claim 19 rejection. For the rest of the features in claim 20, see claim 8 rejection.

register the device object tree with the device class library to enable receipt of actions/events for the services of the root device and the services of the embedded devices; and register an event listener class of the device class library with the UPnP software developing kit to receive action/event requests from one or more control points of the UPnP device that are forwarded to service objects within the device object tree.	
21. The apparatus of claim 19, comprising: one or more root devices, each including one or more services for responding to actions and events received by the respective root device, and one or more embedded devices including one or more services to respond to actions and events received by a respective embedded device.	For the features of claim 19 see claim 19 rejection. For the rest of the features in claim 21, see claim 8 rejection.
22. The apparatus of claim 20, wherein the processor is further caused to: receive a UPnP device description document from a device developer, generate one or more service control class files including one or more service-control stub-method based on the UPnP device description document and one or more service control protocol description files listed within the UPnP device description document, and provide the service control class files including service-control stub-methods to the device developer in order to receive updated service control stub-methods including code for responding to actions and events received by a	For the features of claim 20 see claim 20 rejection. For the rest of the features in claim 22, see claim 8 rejection.

UPnP device described by the UPnP
device description document.

10. Claims 3, 12 are rejected under 35 U.S.C. 103(a) as being unpatentable over US 2002/0112058, by Weisman et al. (hereinafter "Weisman"), in view of US patent No. 6,549,943 by Maximilian J. Spring (hereinafter "Spring"), further in view of 'UPnP Device Architecture', June 2000 (hereinafter "UPnP"), and further in view of US 2002/0035621, by Zintel et al. (hereinafter "Zintel").

CLAIM	**Weisman / Spring / UPnP / Zintel**
3. The method of claim 2, further comprising: (a) generating a root directory for the root device; (b) storing each of the one or more services of the root device within the root directory; (c) selecting an embedded device from the one or more embedded devices of the root device; (d) generating a sub-root directory for the selected embedded device; (e) storing each of the one or more services of the selected embedded device within the sub-root directory; and (f) repeating generating and storing for each of the one or more embedded devices of the root device.	For the features of claim 2 see claim 2 rejection. Weisman and UPnP teach the aspects of claim 3, but does not specifically mention "root directory' (item a). However, Zintel has disclosed this feature in paragraph 532, "UPnP also provides a Directories mechanism to allow discovery to scale...When present, a **directory** will read all incoming **service** requests and respond to them itself. This requires that all **services** (e.g., the **embedded computing device** 900) register with the **directory** so that the **directory** is able to properly answer on their behalf. The **directory** is also responsible for communicating with other directories in order to determine whether the service is available within the local network, the WAN and potentially the Internet." Items b-f apply to claim2 rejection (***for all nested devices***). It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement of the Weisman, Spring, and UPnP's disclosure with the directory mechanism taught by Zintel for the purpose of when introducing a new device into a network to automatically configure so as to connect and interact with other computing devices available on the network. (see Zintel Abstract, lines 5-7).
12. The computer readable storage medium of claim 11, further comprising: generating a root directory for the root device; storing each of the one or more services of the root device within the root directory; selecting an embedded device from the one or more embedded devices of the root device; generating a sub-root directory for	For the features of claim 11 see claim 11 rejection. For the 'root directory' feature see claim 3 rejection; for the rest of the features in claim 12, see claim 2 rejection.

the selected embedded device;
storing each of the one or more services of the selected embedded device within the sub-root directory; and
repeating generating and storing for each of the one or more embedded devices of the root device.

11. Claims 7, 16 are rejected under 35 U.S.C. 103(a) as being unpatentable over US 2002/0112058, by Weisman et al. (hereinafter "Weisman"), in view of US patent No. 6,549,943 by Maximilian J. Spring (hereinafter "Spring"), further in view of 'UPnP Device Architecture', June 2000 (hereinafter "UPnP"), and further in view of 'Universal Plug and Play in Windows XP' by Tom Fout (hereinafter "Fout").

CLAIM	**Weisman/ Spring / UPnP / Fout**
7. The method of claim 1, wherein receiving the service control class files further comprises: displaying the one or more service control stub-methods to the device developer; receiving code from the device developer for implementing the one or more service control stub-methods for responding to actions and events received by the 'UPnP device; and once the code is received, storing the received code within the one or more corresponding service control stub-methods.	For the features of claim 1 see claim 1 rejection. Weisman, Spring, and UPnP have taught all the aspects ('receiving code from the device developer', and 'storing the received code') of claim 7 except the 'displaying the service to the device developer'. However, Fout has taught these features in an analogous art, in Fout, page 19, under 'Presentation', last paragraph "The capabilities of the presentation page (***presentation means 'present' to the user, it implies to be 'displayed' to the user***) are completely specified by the UPnP vendor. To implement a presentation page, a UPnP vendor may wish to use UPnP mechanisms for control and/or events, leveraging the device's existing capabilities. Notice that there is no UPnP Forum element defined in presentation, it is completely up to the vendor!" In UPnP, a 'representation' template is shown on page 15-16, and on page 22, under 2.4 'Description: UPnP Service Template', "By appropriate specification of placeholders, the listing above can be either a UPnP Device Template or a UPnP device description. Recall that some placeholders would be defined by a UPnP Forum working committee (colored *red*), i.e., the UPnP device type identifier, required UPnP services, and required UPnP embedded devices (if any). If these were defined, the listing would be a UPnP Device Template, **codifying** the standard for this type of device. UPnP Device Templates are one of the key deliverables from UPnP Forum working committees." Here we've learned that a template can be displayed for the code developer to define the UPnP device. The entered code is received and stored for later use. It would have been obvious to a person of ordinary skill in the art at the time of the

	invention was made to supplement Weisman, Spring and UPnP's teaching by displaying, receiving and storing the device taught by Fout for the purpose of standardize device (see Fout, page 22, 8th paragraph).
16. The computer readable storage medium of claim 10, wherein receiving the service control class files further comprises: displaying the one or more service control stub-methods to the device developer; receiving code from the device developer for implementing the one or more service control stub-methods for responding to actions and events received by the UPnP device; and once the code is received, storing the received code within the one or more corresponding service control stub-methods.	For the features of claim 10 see claim 10 rejection. For the rest of the features in claim 16, see claim 7 rejection.

Conclusion

The following summarizes the status of the claims:

Claim objections: 2, 19, and any design choice related claims.

35 U.S.C 103 rejections: Claims 1-22

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chih-Ching Chow whose telephone number is 703-305-7205. The examiner can normally be reached on 7:00am - 3:30pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Dam can be reached on 703-305-4552. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Chih-Ching Chow
Examiner
Art Unit 2122

CC

[illegible]

JOHN CHAVIS
PATENT EXAMINER
ART UNIT 2124